CIS 11 Coding Guide

**Option B: Test Score Calculator Program Create an LC-3 program that displays the**

**minimum, maximum and average grade of 5 test scores and display the letter grade associated with the test scores.**

Input:

User is prompt to input the test scores.3

Output:

Display maximum, minimum, average score and letter grade equivalence (0 – 50 = F, 60

– 69 = D, 70 – 79 = C, 80 – 89 = B, 90 – 100 = A) on the console.

The program must fulfill the following criteria:

1.Contain appropriate addresses: origination, fill, array, input and output. (20 points)

2.Display minimum, max, average values/grades in console. (20 points)

3.Use appropriate labels and comments. (20 points)

4.Contain appropriate instructions for arithmetic, data movement and conditional operations.

(40 points)

5. Comprise of 2 or more subroutines and implement subroutine calls. (20 points)

6. Use branching for control: conditional and iterative. (30 points)

7. Manage overflow and storage allocation. (20 points)

8.Manage stack: include PUSH-POP operation on stack. (20 points)

9.Include save-restore operations. (30 points)

10. Include pointer (20 points)

11. Implement ASCII conversion operations (30 points)

12. Use appropriate system call directives. (10 point)

13. Testing (20 points): Test the program using the below values

52

87

96

79

61

**C++ Example: Basic Stack**

7. Manage overflow and storage allocation. (20 points)

8.Manage stack: include PUSH-POP operation on stack.

const int SIZE = 5;

class Stack{

private:

int m\_stack[SIZE];

int m\_top;

public:

Stack(){m\_top = 0;}

void push(int val)

{

if (m\_top >= SIZE) return; // overflow

m\_stack[m\_top] = val;

m\_top += 1;

}

int pop()

{

if (m\_top == 0) return -1;// underflow

m\_top -= 1;

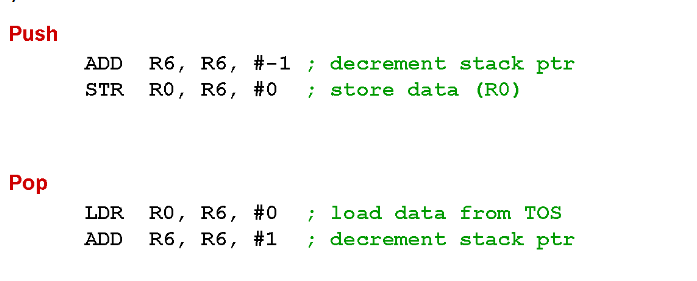
return m\_stack[m\_top];

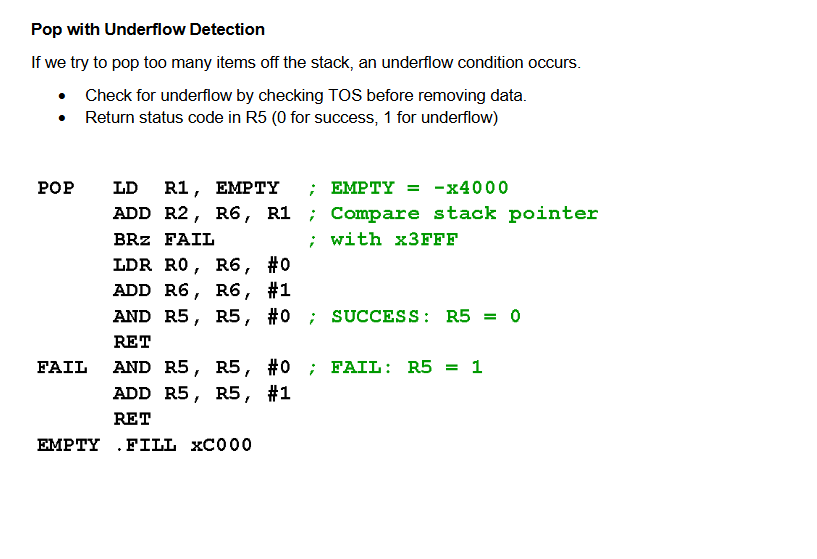
}

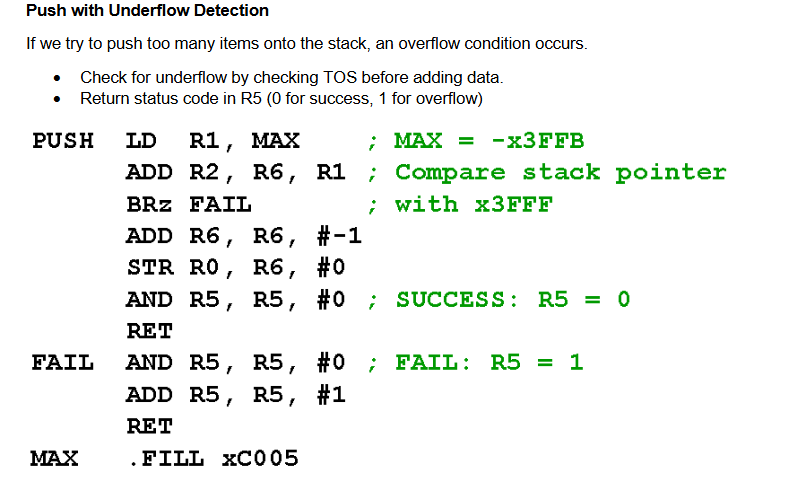
};

**Reference Lectures/Notes: Unit Chapter 9/10**

**LC-3 Code/Psuedocode:**

****

****

****

Display maximum, minimum, average score

**C++ Code Example: Iterative approach to get average, minimum, and maximum amounts**

**double averageAmount (double Total[SIZE])**

**{**

**double total = 0, average = 12;**

**for (int count = 0; count < SIZE; count++)**

**{**

**total += Total[count];**

**}**

**return total/average;**

**}**

**double largestMonth (double Total[SIZE])**

**{**

**double largestTotal = 0;**

**for (int count = 0; count < SIZE; count++)**

**{**

**if (Total[count] > largestTotal)**

**{**

**largestTotal = Total[count];**

**}**

**}**

**return largestTotal;**

**}**

**double smallestMonth (double Total[SIZE])**

**{**

**double smallestTotal = 0;**

**for (int count = 0; count < SIZE; count++ )**

**{**

**if (Total[count] < smallestTotal)**

**{**

**smallestTotal = Total[count];**

**}**

**}**

**return smallestTotal;**

**}**

**LC-3 Pseudo Code/ Code:**

**Input:**

* **5 test scores (0–100) entered by the user**

**C++ Code Example:**

**if (score >= 90) {**

**cout << "A" << endl;**

**}**

**else if (score >= 80) {**

**cout << "B" << endl;**

**}**

**else if (score >= 70) {**

**cout << "C" << endl;**

**}**

**else if (score >= 60) {**

**cout << "D" << endl;**

**}**

**else {**

**cout << "F" << endl;**

**}**

**return 0;**

**}**

**LC-3 PseudoCode/Code:**

**Output:**

* **Maximum Score**
* **Minimum Score**
* **Average Score**
* **Letter Grade:**
  + **A: 90–100**
  + **B: 80–89**
  + **C: 70–79**
  + **D: 60–69**
  + **F: 0–59**

**Expected Output:**

**Max Score: 96**

**Letter Grade A**

**Min Score: 52**

**Letter Grade: F**

**Average: 75**

**Letter Grade: C**